

(19) World Intellectual Property Organization  
International Bureau(43) International Publication Date  
25 January 2001 (25.01.2001)

PCT

(10) International Publication Number  
WO 01/05475 A1

(51) International Patent Classification?: A63F 13/10

(21) International Application Number: PCT/US00/19352

(22) International Filing Date: 14 July 2000 (14.07.2000)

(25) Filing Language: English

(26) Publication Language: English

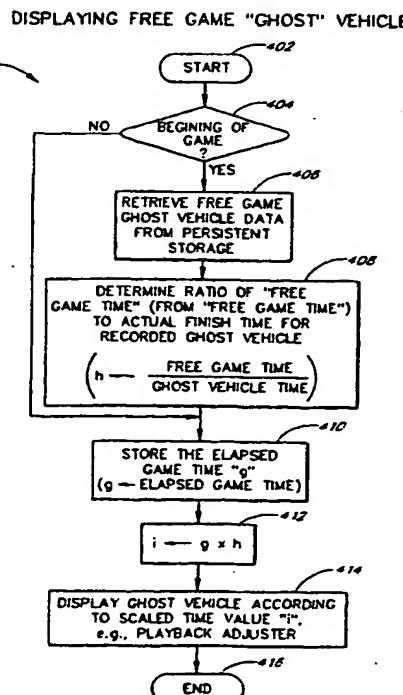
(30) Priority Data:  
60/144,272 15 July 1999 (15.07.1999) US(71) Applicant: MIDWAY GAMES WEST INC. [US/US];  
675 Sycamore Drive, Milpitas, CA 95035-1110 (US).(72) Inventor: HIGHTOWER, Aaron; 340 Arrowhead Way,  
Hayward, CA 94544 (US).(74) Agent: ALTMAN, Daniel, E.; Knobbe, Martens, Olson &  
Bear, LLP, 16th floor, 620 Newport Center Drive, Newport  
Beach, CA 92660 (US).(81) Designated States (national): AE, AG, AL, AM, AT, AU,  
AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ,  
CZ (utility model), DE, DE (utility model), DK, DK (utility  
model), DM, DZ, EE, EE (utility model), ES, FI, FI (utility  
model), GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP,  
KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA,  
MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO,  
RU, SD, SE, SG, SI, SK, SK (utility model), SL, TJ, TM,  
TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.(84) Designated States (regional): ARIPO patent (GH, GM,  
KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian  
patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European  
patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,  
IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG,  
CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— With international search report.

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEM AND METHOD OF VEHICLE COMPETITION WITH ENHANCED GHOSTING FEATURES



(57) Abstract: A system (100) and method (200) for computerized competition useful for rewarding a player. The system and method may be utilized in arcade games, personal computer games, dedicated video games, networked games, and simulators. The method may include selecting a target reward level or threshold (604) such as by selecting a score from a list of past scores, and dynamically adjusting (230/520) the reward level according to the ability of the players of the system. The method may further include adjusting (408) the playback (414) of a previous competition sequence (406) according to the adjusted reward level. In one embodiment, a previous vehicle race sequence is stored (240) and played back as a ghost or phantom vehicle simultaneously (220) with a present vehicle (204) of a player.

WO 01/05475 A1

## SYSTEM AND METHOD OF VEHICLE COMPETITION WITH ENHANCED GHOSTING FEATURES

Background of the Invention5 Field of the Invention

Aspects of the present invention generally relate to vehicle simulators. More particularly, embodiments of the present invention relate to selectively rewarding the user of the simulator.

Description of the Related Technology

10 What is needed is a way to selectively reward users of a simulator or game, such as a vehicle simulator, according to a target reward level established by an owner or operator of the simulator. The target level would vary according to the ability of the users and would be adjusted dynamically.

Summary of the Invention

15

In one aspect of the invention, there is a method of simulated vehicle competition, comprising storing a plurality of parameters indicative of past routes and a past route, providing a threshold route parameter, navigating a simulated vehicle over a current route, displaying the current route of the simulated vehicle, and modifying the threshold route parameter with a parameter corresponding to the plurality of stored parameters.

20

In an additional aspect of the invention, there is a method of rewarding a player of a simulated vehicle racing system, the method comprising a) storing vehicle race parameters of past players on a particular track in a memory, b) storing a target vehicle path, c) selecting one of the stored vehicle race parameters as a target race parameter, d) playing the stored target vehicle path as a function of the target race parameter and a new vehicle path by a player vehicle of a present player, e) recording spatial data of the player vehicle on the particular track in a buffer as the new vehicle path of the present player, f) storing a vehicle race parameter of the present player in the memory, g) selecting the recorded new vehicle path as a new target vehicle path if the stored vehicle race parameter of the present player is an improvement over the target race parameter, h) adjusting the stored vehicle race parameter associated with the new target vehicle path based on the target race parameter, thereby generating a new target race parameter, and i) repeating d) - h) at least one time.

25

In an additional aspect of the invention, there is a simulated vehicle racing method, comprising retrieving a vehicle path corresponding to a stored route of one of a plurality of previous players on a simulated course, retrieving a plurality of vehicle race times, each race time corresponding to a race time of a previous player, selecting one of the plurality of vehicle race times as a free game time, and adjusting the playback of the retrieved vehicle path as a function of the free game time.

5

In an additional aspect of the invention, there is a simulated vehicle system, comprising a simulated vehicle configured to traverse a simulated course, a data structure holding a plurality of course finish times, a present course buffer configured to store a present course path of the simulated vehicle and a course finish time of the simulated vehicle as it traverses the simulated course, a recorded course storage configured to store a recorded course path, and a playback adjuster configured to adjust the speed of playback of the recorded course path when a course finish time in the data structure which corresponds to the recorded course path is different than a selected one of the course finish times.

10

In an additional aspect of the invention, there is a computerized competition method, comprising accumulating a plurality of competition scores from multiple competitions in a competition environment, and selecting one of the competition scores to be a threshold for further competitions, wherein passing the threshold determines an award, and wherein the selecting is based on a predefined percentage of awards.

15

In an additional aspect of the invention, there is a computerized competition system, comprising a competition environment stored in a computer, a data structure storing a plurality of past competition scores, a present competition buffer configured to store a present competition score and results of a present competition in the competition environment, a recorded competition storage configured to store results of a past competition in the competition environment, and a parameter adjuster configured to adjust at least one parameter of playback of the recorded past competition based on a function of a selected one of the competition scores.

20

In an additional aspect of the invention, there is a simulated competition method, comprising retrieving a stored competition sequence of a previous player in a competition environment, retrieving a plurality of scores of previous players, selecting one of the plurality of scores as a target score, and adjusting a playback parameter of the retrieved competition sequence as a function of the target score.

#### Brief Description of the Drawings

Figure 1 is a block diagram of one embodiment of the components for a simulated vehicle competition system of the present invention.

Figure 2 is an operational flowchart describing using the system of Figure 1 for simulated vehicle competition and determining a competition parameter for rewarding the player.

Figure 3A is a flowchart describing one embodiment of the Record Player Vehicle Data function defined in Figure 2.

Figure 3B is a diagram of one embodiment of a data record used in a data structure for storing player vehicle data.

Figure 3C is a diagram of one embodiment of a data structure for storing player vehicle data as described in Figure 3A.

Figure 4 is a flowchart describing one embodiment of the Display Ghost Vehicle function defined in Figure 2.

Figure 5A is a flowchart describing one embodiment of the Free Game Time Adjustment function defined in Figure 2.

Figure 5B is a diagram of one embodiment of a data structure for storing competition data as described in Figure 5A.

5 Figure 6 is a flowchart describing one embodiment of the Update Time to Beat function defined in Figure 5A.

Figure 7 is a flowchart describing one embodiment of the Calculate Time to Beat function defined in Figure 6.

Figure 8 is a flowchart describing one embodiment of the Save Ghost Data function defined in Figure 2.

10 Figure 9 is a diagram of a screen display for an exemplary scene of the vehicle competition system as seen on the display system of the embodiment shown in Figure 1.

#### Detailed Description of the Preferred Embodiments

This present invention incorporates by reference the following U.S. Patents: 5,269,687, entitled "System and Method for Recursive Driver Training"; 5,354,202, entitled "System and Method for Driver Training with Multiple Driver Completion"; 5,366,376, entitled "Driver Training System and Method with Performance Data Feedback"; 5,577,913, entitled "System and Method for Driver Training with Multiple Driver Competition"; and 5,660,547, entitled "Scenario Development System for Vehicle Simulators".

20 The following detailed description presents a description of certain specific embodiments of the present invention. However, the present invention may be embodied in a multitude of different ways as defined and covered by the claims. In this description, reference is made to the drawings wherein like parts are designated with like numerals throughout.

25 One embodiment of the invention is described herein with reference to video game systems in an arcade. However, it will be understood that the invention is applicable to vehicle simulators or game systems of other types as well. For example, the invention may be embodied in home video games, whether played on stand-alone units through game controllers that link to ordinary TV sets or for games that are played on personal computers (PCs). Any of these game units, if provided with suitable hardware and software for accommodating networking operation, could be linked to a global network such as the Internet.

30 Figure 1 illustrates an example of a game system 100 that includes several subsystems, as is known in the art. For example, an arcade game system may include a display system 114 for displaying high resolution, three dimensional images to a screen. The display system 114 communicates with a Central Processing Unit (CPU) system 102 through a bus 116. The bus 116 can be in the form of any conventional data bus, but in one embodiment, is a peripheral component interconnect (PCI) bus as is well known in the electronics technology. The CPU system 102 may include a processor such as a Quantum Effect Design, Inc. R7000 processor or any other well-known processors, such as those provided by Motorola, Hitachi, Intel or IBM. The CPU system 102 may also include an appropriate computer memory 104, an input/output (I/O) subsystem (not shown), and persistent storage 112 for storing computer programs and data. The

memory 104 may include an executable code portion 106 for the vehicle simulation competition and reward system of the present invention and a set of data structures utilized by the code. In one embodiment, the vehicle simulation competition code is embodied as an arcade game. A portion of the source code, written in "C", used to generate one embodiment of the executable code 106 is provided in the Appendix.

5 The CPU system 102 may communicate with the input/output system through a local bus, such as a 32-bit local bus that provides data communications between the processor and the input/output subsystem. Within the input/output subsystem is an I/O connector (not shown) that accepts inputs from peripheral devices or controls 110. In one embodiment, the I/O connector is a Japanese Amusement Machine Manufacturer's Association (JAMMA) connector. This type of connector is well-known in the art of arcade games and provides an interface for determining whether a peripheral 10 event has occurred. For example, in many arcade games, a JAMMA connector is used to determine whether the start, fire, up, down, left or right buttons have been pressed by a player during a game play. Connected to the I/O connector may be a gas pedal, steering wheel, buttons, and so forth. Thus, when action is taken by a player, inputs into the CPU system 102 are activated.

15 In another embodiment, the CPU system 102, the memory 104, the I/O subsystem and the persistent storage 112 may be embodied as a personal computer, such as an IBM-compatible computer or a Macintosh computer from Apple. The personal computer may be connected to a computer network, e.g., the Internet, through an appropriate connection. In yet another embodiment, the CPU system 102, the memory 104, the I/O subsystem and the persistent storage 112 may be embodied in a dedicated game machine, such as available from Nintendo, Sega, Sony, or others.

20 Referring to Figure 2, a top level operational flowchart for a simulated vehicle competition process 200 will be described. Process 200 includes determining a competition parameter for rewarding a player. In one embodiment, the flowchart may describe a car racing arcade game embodied as the executable code 106 (Figure 1). In another embodiment, the flowchart may describe a vehicle simulator or a video game embodied as the executable code 106. Although an arcade game embodiment is described hereinbelow, the description hereinafter also includes vehicle 25 simulator or vehicle competition video game embodiments. Furthermore, the invention may be applied to non-vehicle competition, so long as there is objective criteria to determine when an award should be given. For example, the invention may be embodied in a sports, war, battle, adventure, or other environment, and the objective criteria may be resources remaining, points, treasures gained, or so forth.

30 Beginning at a start state 202, process 200 begins a new vehicle competition game and moves to a Display Player Vehicle function 204. Function 204 displays a simulated vehicle, such as vehicle 900, being driven by a player on a track 950 of a simulated course 940 as shown in the exemplary screen shot of Figure 9. The display of simulated vehicles on a simulated course is well known. For example, Applicant's U.S. Patent No. 5,269,687 and Patent No. 5,366,376 describe displaying a car on a track. A present race time 920 is also displayed, such as in a corner of the display.

At the completion of displaying the player's vehicle 900 at one point of time on the track 950, process 200 advances to a Record Player Vehicle Data function 210. Data regarding the position, orientation and time of the player's vehicle is stored in memory. Function 210 will be further described in conjunction with Figures 3A and 3B.

After the player vehicle data is stored for the instantaneous position of the player vehicle on the course, process 200 continues to a Display "Ghost" Vehicle function 220. In one embodiment, a "ghost" vehicle, such as ghost vehicle 910 (Figure 9), is displayed as a lighter and translucent version of a player vehicle, although there are other ways of showing a ghost vehicle. The ghost vehicle may appear to be similar in size, color and shape, etc. to the player's vehicle, or may be different in size, color or shape, etc. The ghost vehicle may also be referred to as a phantom vehicle. In one embodiment, the word "FREE" may appear over the ghost vehicle as will be described hereinbelow.

After the ghost vehicle 910 is displayed at one point of time on the track 950, process 200 advances to a decision state 222 to determine if the present game has completed. If not, process 200 returns to function 204 to display the player vehicle at a subsequent time to the previous time of display. In one embodiment of the system 100, the loop of functions 204, 210 and 220 is performed about every 16 milliseconds or 60 times a second.

If the present game has completed, as determined at decision state 222, process 200 moves to a Free Game Time Adjustment function 230. In one embodiment of the present invention, a player receives a free game for beating the race finish time of the ghost car 910 (having "FREE" above it) as it traverses the course 940. The system 100 may include a plurality of courses or tracks, each of which may include a ghost car and a corresponding race finish time. The time to beat to win a free game may be adjusted to track a game operator's setting, for example, as will be explained in conjunction with Figures 5A, 5B, 6 and 7 hereinbelow.

After the free game time is adjusted by function 230, process 200 advances to a Save Ghost Data function 240. If the player beat the free game time, the system 100 stores new ghost vehicle data as will be described in conjunction with Figure 8 below. At the completion of function 240, process 200 moves to a decision state 242 to determine if the player desires to play a new game. If so, process 200 moves to start a new game at function 204, as described above. If it is determined, at decision state 242, that a new game is not to be played, process 200 completes at an end state 244.

Referring to Figures 3A and 3B, the Record Player Vehicle Data function 210, previously shown in Figure 2, will be further described. Beginning at a start state 302, function 210 proceeds to a decision state 304 to determine if the player's vehicle has advanced a distance of 30 feet on the simulated course 940 (Figure 9) since the last time player vehicle data was stored. In another embodiment, a distance other than 30 feet may be used, such as a distance selected from the range of 5 to 50 feet, for example. The selected distance value balances two opposing constraints. Using a short distance requires more memory to be used, and using too long a distance causes an undesirable effect to occur, i.e., there may be artifacts in the animation when played back or reproduced. In one embodiment, the distance value may be based on the size of the available memory. In another embodiment, advanced techniques may lower the memory usage and may be used on a less powerful system, such as a home gaming system, for example. Player

vehicle data is stored at the beginning of the course as initial stored data. If the vehicle has not moved more than 30 feet, as determined at decision state 304, function 210 completes and proceeds to end state 308.

If the vehicle has moved more than 30 feet, as determined at decision state 304, function 210 proceeds to state 306. At state 306, the present vehicle data is stored in a vehicle data structure, which functions as a present course buffer. For example, data regarding the position, orientation and game time of the player's vehicle is stored in the data structure 108 in memory 104. An exemplary queue data structure 330 having multiple data records, such as data records 320 and 340, is shown in Figure 3C. In one embodiment, the data record 320 (also for record 340 and subsequent records) has fields 322 for vehicle X, Y, Z position, fields 324 for quaternion q1, q2, q3, q4 orientation, and a game time field 326, and is shown in Figure 3B. Quaternion orientation may be implemented by computer functions such as described in *Advanced Animation and Rendering Techniques, Theory and Practice*, Alan Watt and Mark Watt, Addison Wesley, Menlo Park, California, ISBN 0-201-54412-1, pages 361-368, which is hereby incorporated by reference. Data for an exemplary record is as follows:

X: 3354.7895  
Y: 4521.8561  
Z: 30.5435  
q1: 0.7071  
q2: 0.5000  
q3: 0.0000  
q4: 0.5000  
game time: 32043 (units in milliseconds)

A new record with data corresponding to a distance of 30 feet down the track is stored at location 340 of the structure 330 in a subsequent execution of function 210. After the completion of state 306, function 210 completes at an end state 308.

Referring to Figure 4, the Display Ghost Vehicle function 220, previously shown in Figure 2, will be further described. Beginning at a start state 402, function 220 proceeds to a decision state 404 to determine if this execution of the function 220 is at the beginning of the game. If so, process 220 retrieves ghost vehicle path data, such as stored in a queue data structure 330 (Figure 3B) by function 240, from persistent storage 112 (Figure 1) at state 406. However, if the ghost vehicle path data has already been previously retrieved, as determined at decision state 404, function 220 proceeds to state 408. At state 408, function 220 determines a ratio of the race time needed to win a free game ("free game time"), which is obtained from a Free Game Time function 620 (Figures 6 and 7) that is part of the Free Game Time Adjustment function 230, to the actual finish time for the stored ghost vehicle ("ghost vehicle time"). This ratio is used to set a variable "h". For example, if a free game time was 210,453 milliseconds and a ghost vehicle time was 212,341 milliseconds, the ratio "h" of free game time to ghost vehicle time would be 0.99110864.

Moving to state 410, function 220 sets a variable "g" equal to the elapsed game time. An exemplary elapsed game time may be 32,043 milliseconds. Continuing at state 412, function 220 multiplies the value of variable

"g" and the value of variable "h" and sets the product equal to the variable "i". Using the above exemplary values, "i" may be 31,758. Moving to state 414, function 220 displays the ghost vehicle according to the scaled time value "i". The software code for state 414 operating on the CPU system 102 functions as a playback adjuster in one embodiment. Showing the position and orientation of a recorded path in three-dimensions along with an orientation using quaternions and a rendered three-dimensional object are discussed in *Advanced Animation and Rendering Techniques, Theory and Practice*, mentioned above. State 414 uses the scaled time value "i" to reference as an index into the time-ordered animation data, e.g., ghost vehicle data. Then, known techniques, such as determining a weighted average of two samples based on a linear proximity in time between sample time values, *ibid.*, are used in the display of the ghost vehicle. Therefore, in one embodiment, a single animation may be played at different speeds based on the scaling ratio "h". For example, referring to Figure 3C, the scaled time value "i" may lie between the game time of data record 320 and the game time of data record 340. In one embodiment, an interpolation is done to calculate new X, Y, Z, q1, q2, q3 and q4 values based on the proximity in time of the scaled time value "i" between the game times of the two data records. Function 220 completes at an end state 416.

Referring to Figures 5A and 5B, the Free Game Time Adjustment function 230, previously shown in Figure 2, will be further described. In one embodiment, function 230 is performed after completion of a game, i.e., after the present game ends but before another game may begin. Portions of the software code and the CPU system 102 (Figure 1) function as a parameter adjuster.

Beginning at a start state 502, function 230 proceeds to a decision state 504 to determine if the player's vehicle finished the present race. If not, function 230 moves to state 506 and searches for the worst value of the competition parameter, such as slowest time, in a data structure 530 containing one or more competition parameters, such as race finish times. The worst finish time is set to a variable "a". In one embodiment, the data structure 530 may be a circular queue of race finish times, scores, etc. The circular queue 530 is one of the data structures 108 (Figure 1). In one embodiment, the circular queue 530 contains one hundred entries. In another embodiment, the circular queue 530 may have a number of entries that is greater or less than one hundred.

Returning to decision state 504, if the player did finish the race, function 230 proceeds to state 508 and sets the variable "a" to the finish time for the race. At the completion of either state 508 or state 506, function 230 advances to state 510. At state 510, function 230 stores variable "a" into a data structure, such as the circular queue 530. Referring to Figure 5B, the slot in the circular queue which just receives an entry, such as the variable "a", is referred to as the tail 532 of the queue. The location just in front of the tail is referred to as the head 534 of the queue.

At the completion of state 510, function 230 proceed to a decision state 512 to determine if the finish time is good enough for a free game. This may happen if the player's vehicle beats the ghost car across the finish line for the course. If so, function 230 continues at state 514 and increments a count of free games awarded by the game system 100. At the completion of state 514, or if the finish time was not good enough for a free game, as determined at decision state 512, function 230 proceeds to state 516. At state 516, function 230 increments a total number of

game played on the system 100. Function 230 then proceeds to an Update Free Game Time function 520 which is described below. At the completion of function 520, function 230 completes at an end state 522.

Referring to Figure 6, the Update Free Game Time function 520, previously shown in Figure 5a will be further described. Beginning at a start state 602, function 520 proceeds to state 604 where a percentage of free games to allow for the game system 100 is used to set a variable "b". In one embodiment, the values for "b" may be 5%, 10% or 15% free games. The percentage of free games to allow has previously been selected by the operator, owner or administrator of the system 100. In another embodiment, the percentage of free games to allow may be initially set by the manufacturer, i.e., the factory. Variable "b" is referred to as the operator's choice percentage.

Moving to state 606, an observed percentage based on a number of free games awarded for previously played games and a total number of previously played games, which are tracked by the game system 100, is used to set a variable "c". This variable is referred to as the observed percentage. Continuing to a decision state 608, function 520 determines if the observed percentage is greater than the operator's choice percentage. If not, function 520 sets the value of variable "b" to a variable "d" at state 610. However, if the observed percentage of free games is greater than the operator's choice percentage, as determined at decision state 608, function 520 proceeds to state 612. At state 612, function 520 calculates a value for variable "d" by subtracting the value of variable "b" from the value of variable "c" and taking that result and subtracting it from the value of variable "b". This result is used to set the variable "d". For example, for the exemplary values of "b" = 5% and "c" = 5.1%, "d" would be  $(b-(c-b)) = 4.9\%$ .

Advancing to a decision state 614, function 520 determines if the value of the variable "d" is less than zero. If so, function 520 proceeds to state 616 and sets the variable "d" to be equal to zero. In one embodiment, the effect of setting "d" to zero forces the Calculate Free Game Time function 620 (Figure 7) to return the fastest time in the circular queue 530 as the new free game time (time to beat), as is discussed below. At the completion of state 610 or state 616, or if the value of variable "d" is not less than zero, as determined at decision state 614, function 520 proceeds to function 620. Function 620 calculates a free game time as a function of the variable "d" and of the circular queue of times 530. Function 620 will be described below in conjunction with Figure 7. Note that the free game time is shown on the exemplary screen shot of Figure 9 as the Free Game Time 930. At the completion of function 620, function 520 finishes at an end state 622.

Of course, other techniques could be used to update the free game time, especially when it is determined that statistical game times deviate from the intended level of awarding free games.

Referring to Figure 7, the Calculate Free Game Time function 620, previously shown in Figure 6, will be further described. Beginning at a start state 702, function 620 proceeds to state 704. At state 704, function 620 transfers the values of the circular queue 530 to a temporary array "e". If the queue is not full, the values are transferred into the top of the array. Advancing to state 706, function 620 sorts temporary array "e" in ascending order of the competition parameter, which may be the race finish times. In one embodiment, this state yields an array of one hundred locations with the fastest finish time stored in location zero and the slowest finish time stored in location 99. Moving to state 708, function 620 obtains an entry in temporary array "e" corresponding to the

percentage variable "d". In operation, since the circular queue 530 and thus temporary array "e", may not always be full, an interpolation between existing elements in the queue may be done. Note that interpolation may also be done on a full array. Also note that if the percentage variable "d" is fractional, multiple entries may be used for interpolation to obtain a more precise value.

5 For example, the value of "d" may be 4.9% and the sorted temporary array "e" may have the following ten values:

<u>element</u>	<u>value</u>
0:	230747
1:	231001
2:	231567
3:	231588
4:	231666
5:	232189
6:	234438
7:	237069
8:	239421
9:	241597

10 20 In one embodiment, the determination of the free game time may be done as follows: First, a fractional number representing the index of the number or numbers to use and a fractional interpolation value to use to interpolate between two numbers is determined using the following equation:

"d" (in percentage) \* < number of elements in array "e" > / 100%.

25 For the value of "d" = 4.9%, and the number of elements in array "e" = 10:

4.9% \* 10 (total entries in list) / 100% = 49 / 100 = 0.49

30 The index of the first element to use is the integer part of 0.49 (which is 0) and the fractional part is used to interpolate between elements 0 and 1. Therefore, the free game time would be:

$$\begin{aligned}
 & e[0] + (e[1] - e[0]) * \text{frac} \\
 & 230747 + (231001 - 230747) * 0.49 - \\
 & 230747 + (254) * 0.49 - \\
 & 230747 + 124.46 - \\
 & 230871 \text{ milliseconds, or } 3'40"871 \text{ (3 minutes, 50 seconds, and 871 milliseconds)}
 \end{aligned}$$

At the completion of state 708, function 620 completes at an end state 710.

40 Referring to Figure 8, the Save Ghost Data function 240, previously shown in Figure 2, will be further described. Beginning at a start state 802, function 240 proceeds to a decision state 804 to determine if the player beat the free game time for the present race. If not, function 240 completes and proceeds to an end state 808. However, if the player did beat the free game time, as determined at decision state 804, function 240 moves to state

806. At state 806, function 240 stores the ghost data, which was temporarily stored to a buffer queue in function 210, to the persistent storage 112, which in one embodiment, functions as a recorded course storage. The buffer data, which is from the queue data structure 330, is supplemented with the race finish time for the course and the name of the player that traversed the course. At the completion of state 806, the function 240 proceeds to an end state 808.

5

The example discussed above relates to a game wherein the parameter that is compared is time, and the reward is a free game. Other embodiments, such as a simulator, may use different parameters for comparison, such as a count of accidents, a score, etc., and other types of rewards. For example, the competition system may be connected to a global computer network and the reward may comprise recognition for the player on a network resource of the global computer network. As another example, the competition system may be connected to a global computer network and the reward may comprise cyber-credits or a cyber-reward for the player on the global computer network.

10

In another embodiment, the competition system may be connected to a global computer network and the vehicle race (ghost) data may be transmitted to a network resource. The vehicle race data for a particular player may be compiled with the vehicle race data of other players. Players who access the network resource may be able to download the vehicle race data to use as ghost data in their competition system.

15

Specific blocks, sections, devices, functions and modules may have been set forth. However, a skilled technologist will realize that there are many ways to partition the system of the present invention, and that there are many parts, components, modules or functions that may be substituted for those listed above.

20

While the above detailed description has shown, described, and pointed out the fundamental novel features of the invention as applied to various embodiments, it will be understood that various omissions and substitutions and changes in the form and details of the system illustrated may be made by those skilled in the art, without departing from the intent of the invention.

## 1. APPENDIX

```
/*
 *
5 -----
-----
*
* $RCSfile: freeg.c,v $ -
*
10 * Copyright (C)1999 Atari Games
* Unauthorized reproduction, adaptation, distribution, performance or
* display of this computer program or the associated audiovisual work
* is strictly prohibited.
*
15 *
-----
-----
*
* $Author: hightower $      $Date: 1999/07/13 05:16:38 $
* $Revision: 1.4 $          $Locker: $
20 *
-----
-----
*
* /
*
25 #include "freeg.h"
#include "hi.h"
#include "cksum.h"
#include "game.h"
*
30 #include "defines.h"
*
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
*
35 #define FG_FILEVERSION 1
*
*
40 * NOTE: this module uses PLAYERS/team.rwm even though it should have it's
*       file named something like "FREEGAME/game_history.rwm"
*       Apologies in advance for any confusion.. (AMH)
* /
45 #define CONVERT(time,laps) (((time)*3)/(laps))
#define UNCONVERT(time,laps) (((time)*laps)/(3))
*
50 static BOOL fg_read;
*
typedef struct _fgTrackHist
{
    /* head--tail -> empty
     * head--tail+1 -> 1 item in list at samp[tail]
```

```

    • head == tail-1 -> list full
    •
    • head > = FG_MAXSAMPS -> undefined ..not allowed
    • tail > = FG_MAXSAMPS -> undefined ..not allowed
    • */
5   S32 head;
    S32 tail;
    S32 samps[FG_MAXSAMPS];

10  S32 curttb;
    U32 totalgames; /* Count of free games */
    U32 freegames; /* total game count */
    }
    fgTrackHist;
15  typedef struct _fgDb
    {
        fgTrackHist h[FG_MAXTRACKS][FG_MODES];
    }
20  fgDb;
    static void
    _insert( fgTrackHist *h, S32 val )
    {
25    h->samps[h->head ++] = val;
    if( h->head > = FG_MAXSAMPS ) h->head = 0;

        if( h->head == h->tail )
        {
30        h->tail++;
        if( h->tail > = FG_MAXSAMPS ) h->tail = 0;
        }
    }

35  static fgDb fg;
    static S32 _fgTime2Beat( S32 track, S32 mode, S32 percent );
    static void _fgRead( void );

    /* Race time in units of milliseconds per 3-laps */
40  static S32 ttb[FG_MAXTRACKS][FG_MODES] =
    {
        { 140000, 130000 }, /* T1 */
        { 193000, 190000 }, /* T2 */
        { 193000, 190000 }, /* T3 */
45        { 194000, 190000 }, /* T4 */
        { 265000, 260000 }, /* T5 */
    };
    void
50  fgInit( void )
    {
        _fgRead();

```

```

        }

5     static void
        _fgUpdateTimes( void )
    {
        F32 ratio;
        S32 i,j;

10    for( i=0; i< FG_MAXTRACKS; i++ )
    {
        for( j=0; j< FG_MODES; j++ )
        {
            S32 val;
            fgTrackHist *h;

15    h = &fg.h[i][j];

            ratio = gDstWinRatio;

20    if( h->totalgames )
    {
        F32 oratio; /* observed ratio */

25    oratio = 100.0f * h->freegames / h->totalgames;

30    if( oratio > ratio )
    {
        ratio = ratio + ( oratio - ratio );
        if( ratio < 0 ) ratio = 0;
    }
    val = _fgTime2Beat( i, j, ratio );

35    if( val ) h->curttb = val;
    else
    {
        h->tail = 0;
        h->head = 1;
        h->samps[0] = h->curttb - ttb[i][j];
    }
40    }
    }
45    static S32
        fgWorst( S32 tidx, S32 midx )
    {
        fgTrackHist *t;
        S32 i;
        S32 worst;

50    t = &fg.h[tidx][midx];

```

```

        if( t->tail == t->head ) return ttb[tidx][midx];

        worst = t->samps[t->tail];
5
        for( i=t->tail; i!=t->head; i=((i+1)%FG_MAXSAMPS) )
        {
            if( t->samps[i] > worst ) worst = t->samps[i];
        }
10
        return worst;
    }

/*
15 * Return the percent of finish times that are better than the given time
 */
S32
    fgPercent( S32 track, S32 mode, S32 time, S32 laps )
{
20
    fgTrackHist *t;
    S32 better = 0;
    S32 total = 0;
    S32 i;
    S32 val = CONVERT(time,laps);

25
    t = &fg.h[track][mode];

    for( i=t->tail; i!=t->head; i=((i+1)%FG_MAXSAMPS) )
    {
30
        if( t->samps[i] < val ) better++;
        total++;
    }

35
    return ( total == 0 ) ? 0 : ( better * 100 ) / total;
}

static int
comparS32( const void * _a, const void * _b )
{
40
    S32 *a = ( S32 * ) _a;
    S32 *b = ( S32 * ) _b;

    return *a - *b; /* Sort in ascending order */
}

45
S32
    fgTime2Beat( S32 track, S32 mode, S32 percent )
{
50
    S32 i;
    S32 total = 0;
    S32 tmp[FG_MAXSAMPS] = {0};
    F32 fidx;

```

```

F32 frac;
fgTrackHist *t;

5   if( percent > 100 ) percent = 100;
   if( percent < 0 ) percent = 0;

   t = &fg.h[track][mode];

10  for( i=t->tail; i!=t->head; i=(i+1)%FG_MAXSAMPS )
{
   tmp[total++] = t->samps[i];
}

15  qsort( tmp, total, sizeof( S32 ), comparS32 );
if( total > 0 )
{
   fidx = ( (F32)percent * (F32)total ) / ( 100.0f );

20  if( fidx >= total ) fidx = total - 1;
}
else fidx = 0;

25  i = (S32)fidx;
frac = fidx * (F32)i;

if( i+1 < total )
{
   30  return tmp[i] + ( tmp[i+1] - tmp[i] ) * ( frac );
}
else
{
   35  return tmp[i];
}
}

BOOL
fgEarned( S32 track, S32 mode, S32 time, S32 laps )
{
40  BOOL earned;
fgTrackHist *h;

if( !fg_read ) _fgRead();

45  if( !gDstWinRatio ) return 0;
if( !laps ) return 0;
if( track < 0 || track > FG_MAXTRACKS ) return 0;
if( mode < 0 || mode > FG_MODES ) return 0;
if( !time ) return 0;

50  h = &fg.h[track][mode];

```

```

    earned = ( CONVERT(time,laps) < h-> curttb );

    return earned;
}

5
static void
_fgWrite( void )
{
    FILE *f;
10
    ckSum ck;
    S32 ver = FG_FILEVERSION;

    if( ( f = fopen( "/d0/PLAYERS/teams.rwm", "r+" ) ) )
    {
15
        ck = ckSumGet( &fg, sizeof( fg ) );

        fwrite( &ver, sizeof( ver ), 1, f );
        fwrite( &ck, sizeof( ck ), 1, f );
        fwrite( &fg, sizeof( fg ), 1, f );
20
        fclose( f );
    }
}

25
static void
_fgRead( void )
{
    FILE *f;
    ckSum ck1;
    ckSum ck2;
30
    S32 ver = FG_FILEVERSION;

    if( fg_read ) return;

    bzero( ( void * ) &fg, sizeof( fg ) );

    if( ( f = fopen( "/d0/PLAYERS/teams.rwm", "r" ) ) )
    {
40
        fread( &ver, sizeof( ver ), 1, f );
        if( ver == FG_FILEVERSION )
        {
            fread( &ck1, sizeof( ck1 ), 1, f );
            fread( &fg, sizeof( fg ), 1, f );
45
        }
        fclose( f );
    }

50
    ck2 = ckSumGet( &fg, sizeof( fg ) );

    if( !ckSumEqual( ck1, ck2 ) )

```

```

    {
        bzero( ( void * ) &fg, sizeof( fg ) );
    }

5      _fgUpdateTimes();

    fg_read = 1;
}

10
/*
 * Submit a race time for entry in the database for determining the
 * race time to beat
 */
15 void
fgSubmit( S32 track, S32 mode, S32 time, S32 laps )
{
    BOOL earned;
    fgTrackHist *h;
20
    h = &fg.h[track][mode];

    if( !laps ) return;
    if( track < 0 || track > = FG_MAXTRACKS ) return;
25    if( mode < 0 || mode > FG_MODES ) return;

    h = &fg.h[track][mode];

    /* NOTE: if they don't finish the race, make it easier */
30    if( !time )
    {
        _insert( h, _fgWorst(track,mode) );
    }
    else
35    {
        _insert( h, CONVERT(time,laps) );
    }

    earned = fgEarned( track, mode, time, laps );
40
    h->totalgames++;
    h->freegames += earned;

    if( h->totalgames > = 0xffffffff )
45    {
        h->totalgames /= 2;
        h->freegames /= 2;
    }

50      _fgUpdateTimes();

    _fgWrite();

```

```
    }

S32
5    fgTime2Beat( S32 track, S32 mode, S32 laps )
{
    S32 t;

    if( !laps ) return;
    if( track < 0 || track > = FG_MAXTRACKS ) return;
    if( mode < 0 || mode > FG_MODES ) return;

10   t = fg.h[track][mode].curttb;

    t = UNCONVERT(t,laps);

15   return t;
}
```

**WHAT IS CLAIMED IS:**

1. A method of simulated vehicle competition, comprising:
  - 5 storing a plurality of parameters indicative of past routes and a past route;
  - providing a threshold route parameter;
  - navigating a simulated vehicle over a current route;
  - displaying the current route of the simulated vehicle; and
  - modifying the threshold route parameter with a parameter corresponding to the plurality of stored parameters.
- 10 2. The method defined in Claim 1, additionally comprising simultaneously displaying the past route of the simulated vehicle with the current route of the simulated vehicle.
- 15 3. The method defined in Claim 1, wherein modifying the threshold route parameter includes selecting a parameter from the plurality of stored parameters as a function of a factory generated statistic.
4. The method defined in Claim 1, wherein modifying the threshold route parameter includes:
  - 20 sorting the plurality of stored parameters into a sequential order; and
  - selecting a parameter from the sorted plurality of stored parameters at a location determined by a percentage of the total number of stored parameters.
5. The method defined in Claim 1, wherein the threshold route parameter is a free game time.
6. A method of rewarding a player of a simulated vehicle racing system, the method comprising:
  - 25 a) storing vehicle race parameters of past players on a particular track in a memory;
  - b) storing a target vehicle path;
  - c) selecting one of the stored vehicle race parameters as a target race parameter;
  - d) playing the stored target vehicle path as a function of the target race parameter and a new vehicle path by a player vehicle of a present player;
  - e) recording spatial data of the player vehicle on the particular track in a buffer as the new vehicle path of the present player;
  - 30 f) storing a vehicle race parameter of the present player in the memory;
  - g) selecting the recorded new vehicle path as a new target vehicle path if the stored vehicle race parameter of the present player is an improvement over the target race parameter;
  - h) adjusting the stored vehicle race parameter associated with the new target vehicle path based on the target race parameter, thereby generating a new target race parameter; and
  - 35 i) repeating d) - h) at least one time.

7. The method defined in Claim 6, additionally comprising sorting the vehicle race parameters according to a predetermined criteria.

5 8. The method defined in Claim 6, wherein the vehicle race parameter comprises a time to finish a race.

9. The method defined in Claim 6, additionally comprising awarding the player a reward if the stored vehicle parameter of the present player is an improvement over the target race parameter.

10 10. The method defined in Claim 9, wherein the vehicle racing system is an arcade game and the reward comprises a free race.

15 11. The method defined in Claim 9, wherein the vehicle racing system is connected to a global computer network and the reward comprises recognition for the player on a network resource of the global computer network.

12. The method defined in Claim 9, wherein the vehicle racing system is connected to a global computer network and the reward comprises cyber-credits or a cyber-reward for the player on a global computer network.

20 13. The method defined in Claim 6, additionally comprising:  
tagging each stored vehicle race parameter with a date of storage and a time of storage; and  
removing a vehicle race parameter tagged with the oldest date and time from the memory when a new vehicle race parameter is added to the memory.

25 14. The method defined in Claim 6, wherein a portion of the memory storing the vehicle race parameters is organized as a data structure comprising a circular queue.

30 15. The method defined in Claim 14, wherein the circular queue has a length equal to a predetermined number.

16. The method defined in Claim 14, additionally comprising overwriting an oldest entry in the circular queue when a new vehicle race parameter is added.

35 17. The method defined in Claim 6, wherein the target race parameter is modified over time.

18. The method defined in Claim 9, wherein adjusting comprises:

determining a first percentage of rewards to be awarded over time;  
determining a second percentage of rewards already awarded based on previous races; and  
selecting one of the stored vehicle race parameters as a function of the first percentage and the second percentage.

5

19. The method defined in Claim 9, wherein the reward comprises a free game.

20. The method defined in Claim 6, wherein the vehicle race parameters of a particular track are stored in the memory for a predetermined number of previous races.

10

21. A simulated vehicle racing method, comprising:

retrieving a vehicle path corresponding to a stored route of one of a plurality of previous players on a simulated course;

retrieving a plurality of vehicle race times, each race time corresponding to a race time of a previous player;

selecting one of the plurality of vehicle race times as a free game time; and

adjusting the playback of the retrieved vehicle path as a function of the free game time.

15

22. The method defined in Claim 21, wherein adjusting the playback of the retrieved vehicle path is also a function of a vehicle race time corresponding to the retrieved vehicle path.

20

23. A simulated vehicle system, comprising:

a simulated vehicle configured to traverse a simulated course;

a data structure holding a plurality of course finish times;

25

a present course buffer configured to store a present course path of the simulated vehicle and a course finish time of the simulated vehicle as it traverses the simulated course;

a recorded course storage configured to store a recorded course path; and

a playback adjuster configured to adjust the speed of playback of the recorded course path when a course finish time in the data structure which corresponds to the recorded course path is different than a selected one of the course finish times.

30

24. The system defined in Claim 23, wherein the playback adjuster adjusts the speed of playback of the recorded course path to the selected one of the course finish times.

35

25. The system defined in Claim 23, wherein the data structure comprises a circular queue.

26. The system defined in Claim 23, wherein each one of the present course path and the recorded course path includes parameters for a position and an orientation of the simulated vehicle.

27. The system defined in Claim 23, wherein the recorded course path is displayed simultaneously with the present course path.

28. The system defined in Claim 23, wherein the playback adjuster adjusts the speed of playback of the recorded course path according to the ratio of the selected one of the course finish times to the course finish time corresponding to the recorded course path.

10 29. The system defined in Claim 23, wherein the data structure holds a preselected number of course finish times.

15 30. The system defined in Claim 23, wherein the recorded course storage stores a previous present course path as the recorded course path.

31. The system defined in Claim 23, wherein the present course path is stored in the recorded course storage as the recorded course path if the course finish time corresponding to the present course path is faster than the course finish time corresponding to the recorded course path.

20 32. The system defined in Claim 31, wherein the selected one of the course finish times changes over time as additional instances of traversing the simulated course are performed by users of the system.

25 33. The system defined in Claim 32, wherein an instance of traversing the simulated course by a user of the system comprises a game, and the user is awarded a free game if the course finish time corresponding to the present course path is faster than the course finish time corresponding to the recorded course path.

34. The system defined in Claim 33, wherein the selected one of the course finish times changes over time based on a relationship between a percentage of free games to allow and a percentage of prior free games.

30 35. A computerized competition method, comprising:  
accumulating a plurality of competition scores from multiple competitions in a competition environment; and  
selecting one of the competition scores to be a threshold for further competitions, wherein passing the threshold determines an award, and wherein the selecting is based on a predefined percentage of awards.

35 36. A computerized competition system, comprising:  
a competition environment stored in a computer;

a data structure storing a plurality of past competition scores;

a present competition buffer configured to store a present competition score and results of a present competition in the competition environment;

a recorded competition storage configured to store results of a past competition in the competition environment; and

a parameter adjuster configured to adjust at least one parameter of playback of the recorded past competition based on a function of a selected one of the competition scores.

37. The system defined in Claim 36, wherein the at least one parameter of playback is adjusted if one plurality of past competition scores corresponding to the stored results of a past competition is different than a one of the plurality of past competition scores.

38. The system defined in Claim 36, wherein the selected one of the plurality of past competition scores is selected by an owner or operator of the competition system.

### 39. A simulated competition method, comprising:

## **retrieving a stored competition sequence of a previous player in a competition environment:**

retrieving a plurality of scores of previous players:

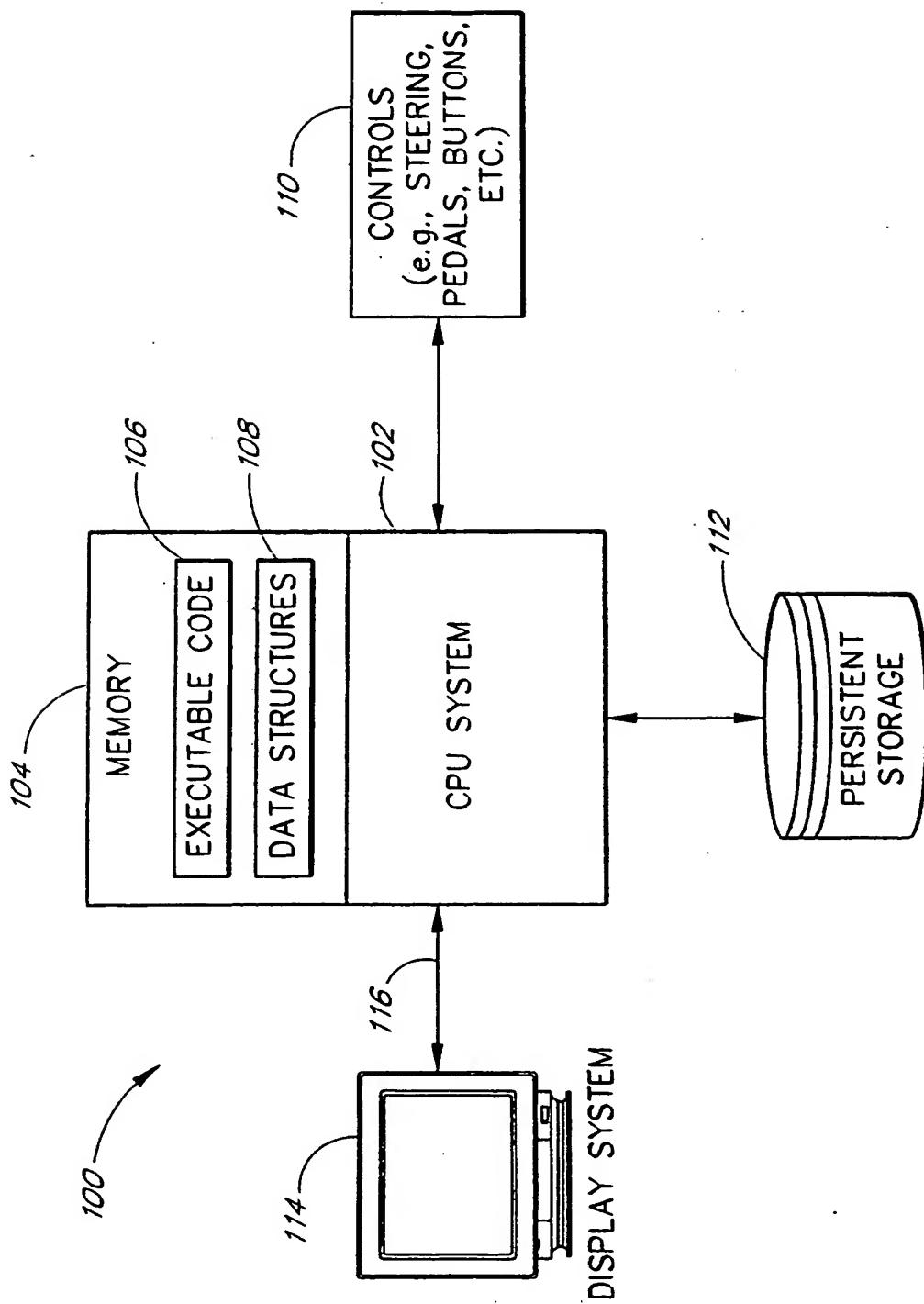
selecting one of the plurality of scores as a target score; and

adjusting a playback parameter of the retrieved competition sequence as a function of the target

40. The system defined in Claim 39, wherein the playback parameter is time.

25

1/11



2/11

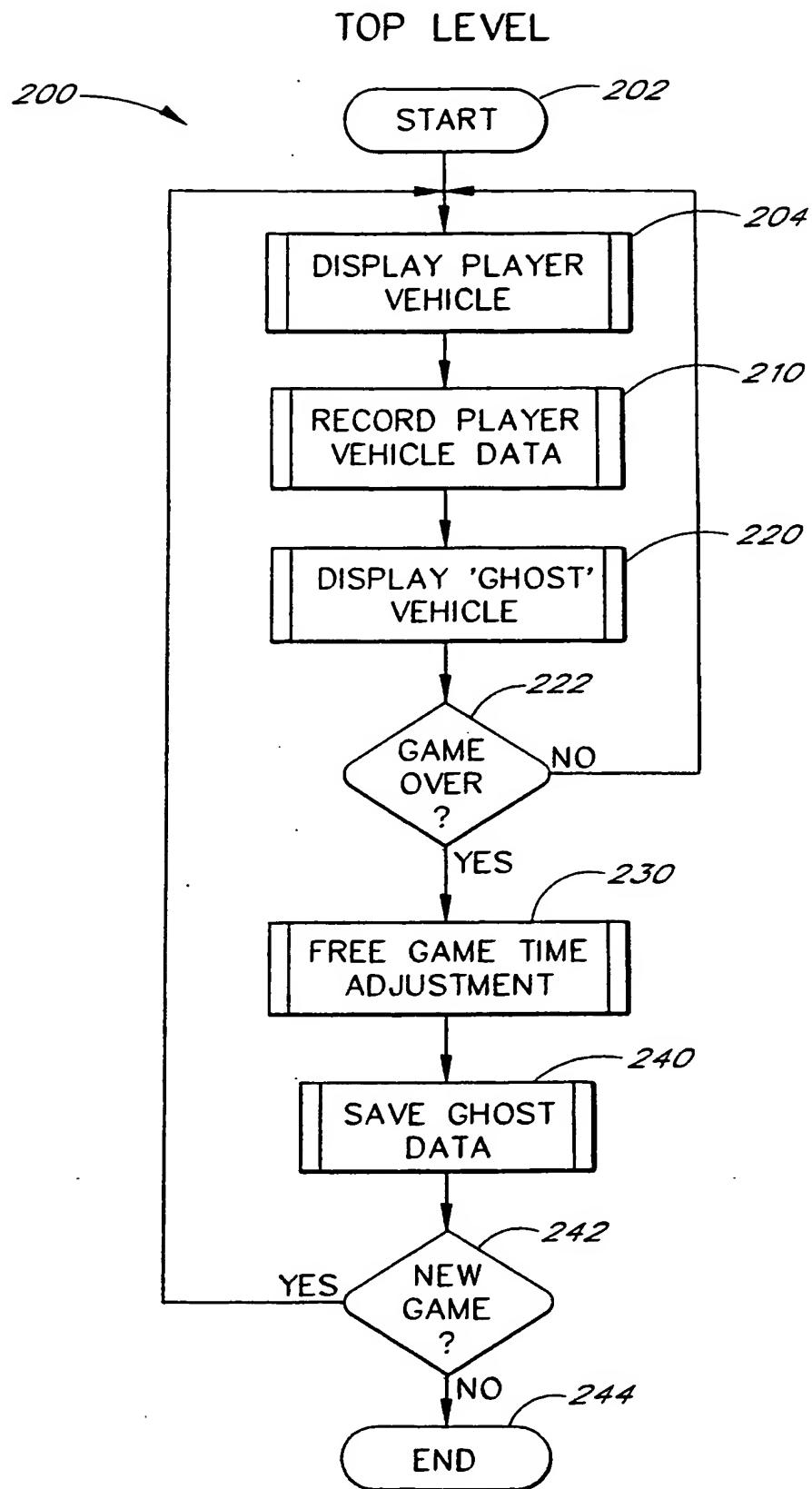


FIG. 2

3/11

## RECORDING PLAYER VEHICLE DATA

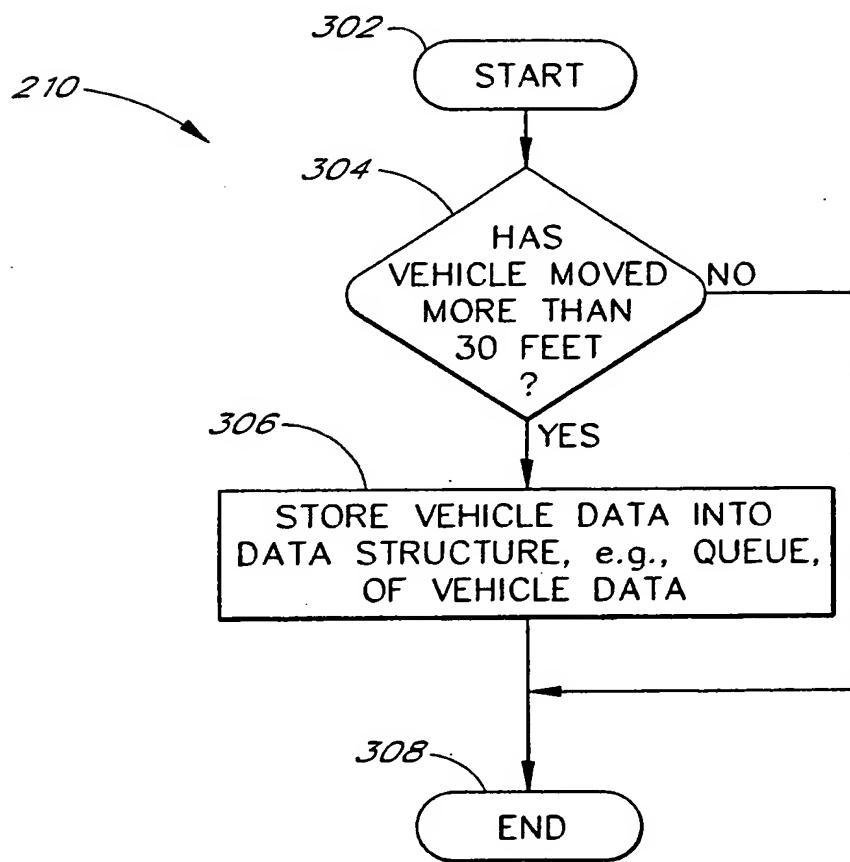


FIG. 3A

4/11

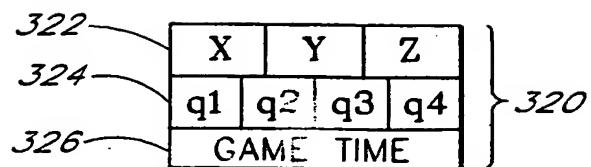


FIG. 3B

330

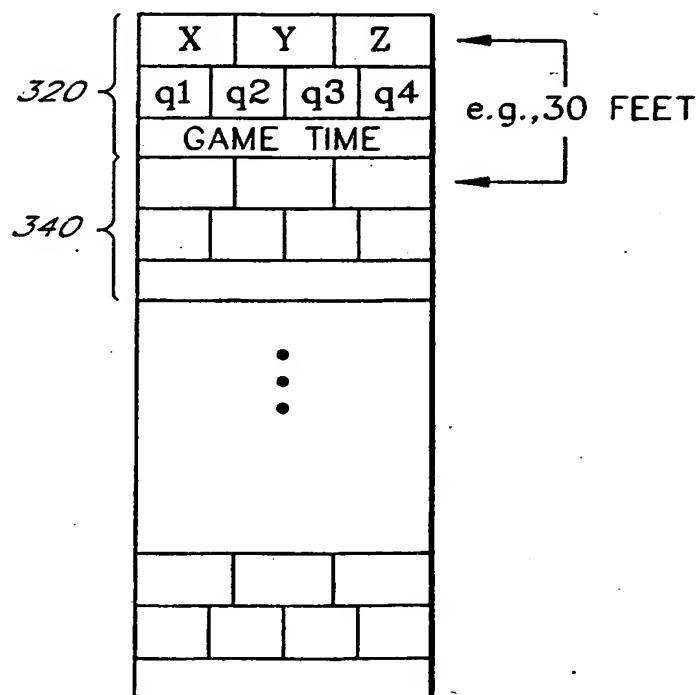


FIG. 3C

5/11

## DISPLAYING FREE GAME "GHOST" VEHICLE

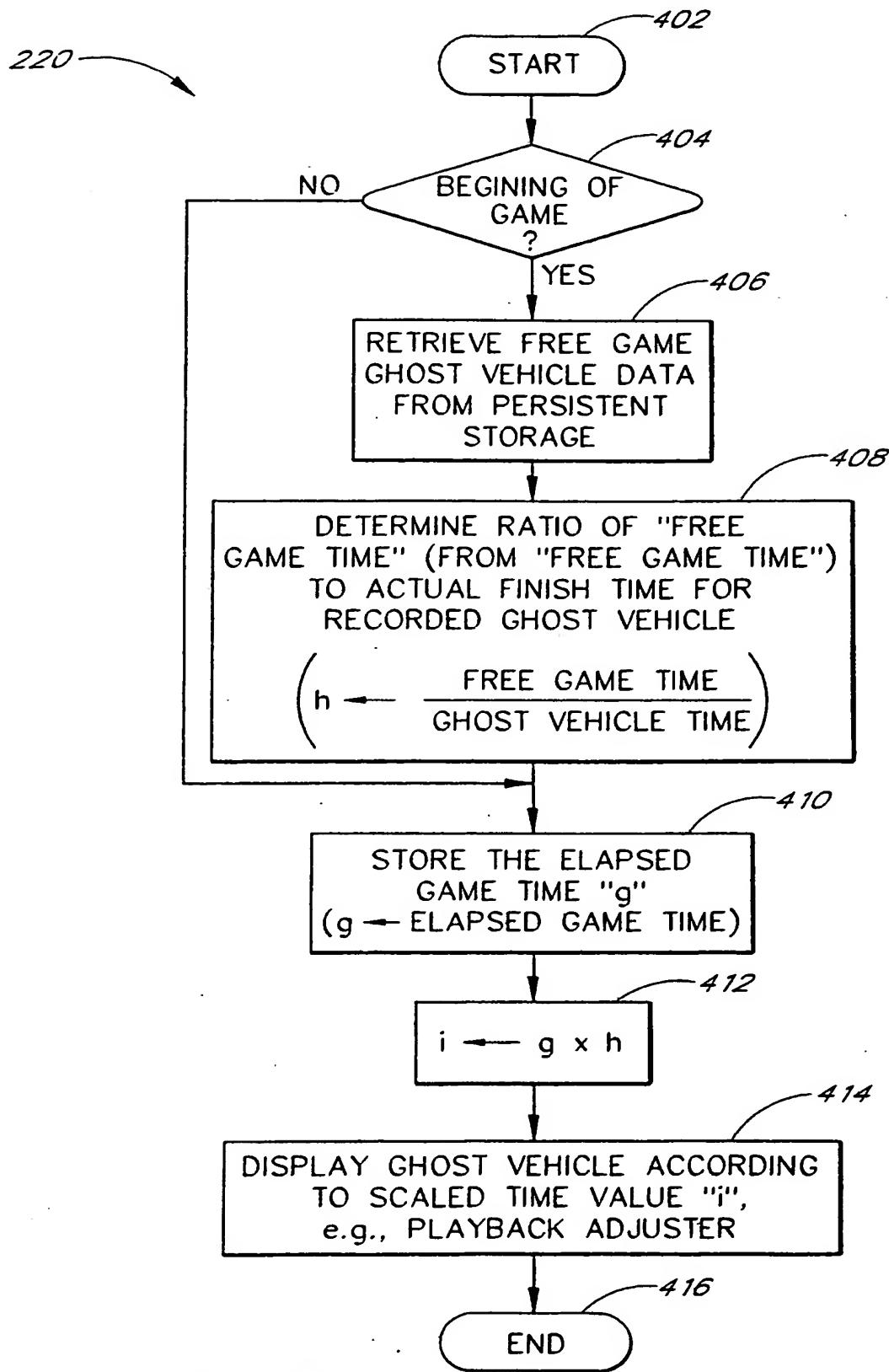


FIG. 4

6/11

## FREE GAME TIME ADJUSTMENT

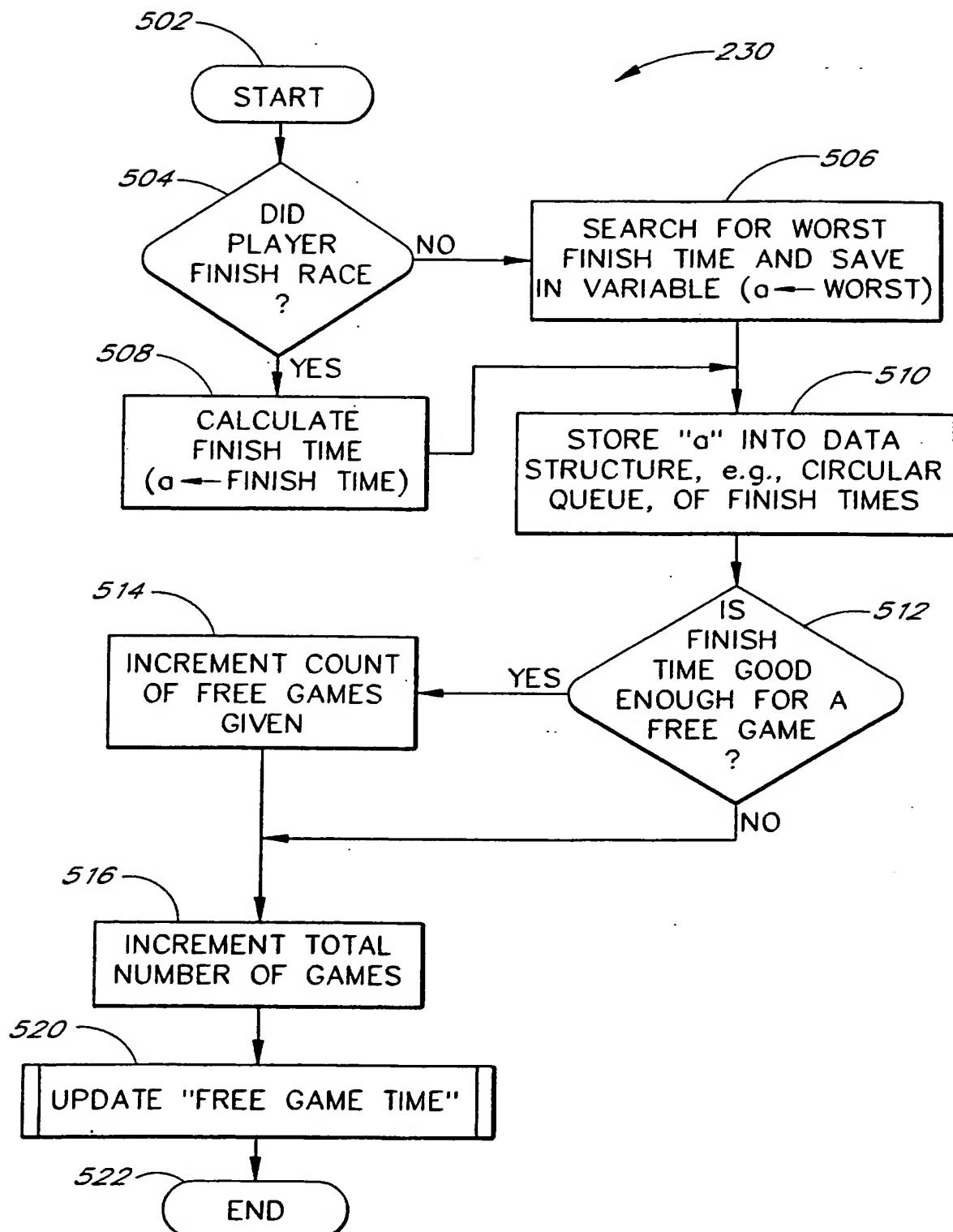
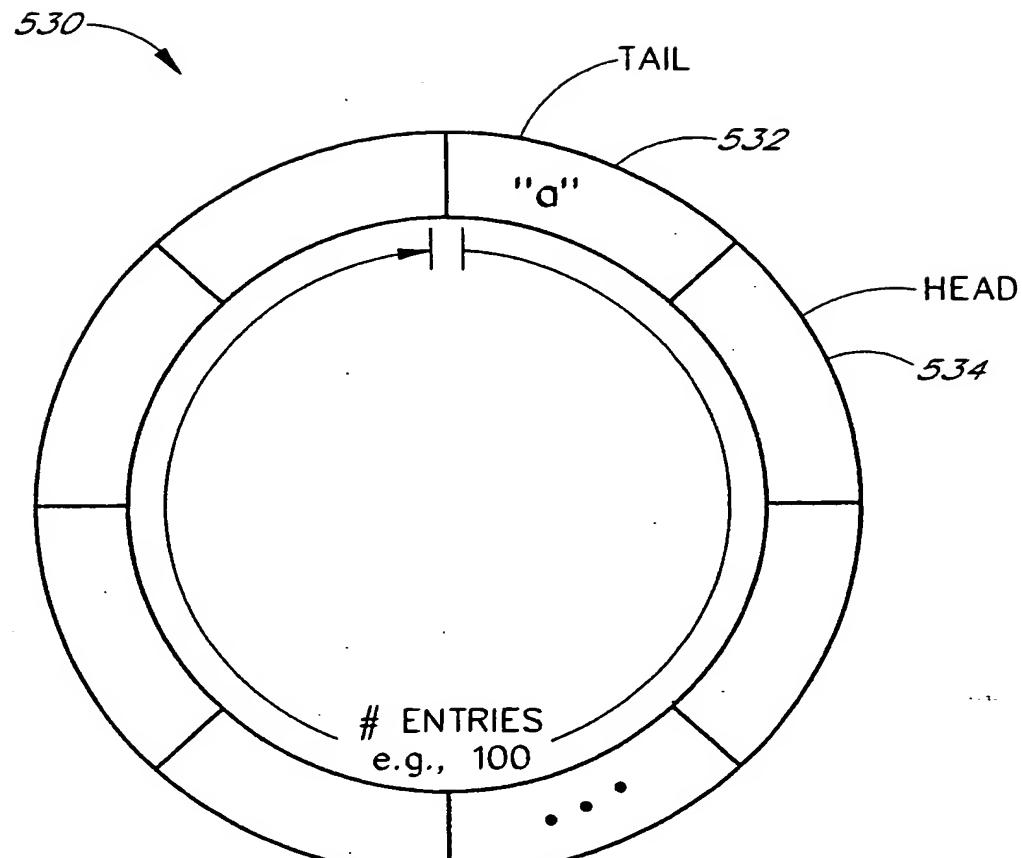


FIG. 5A

7/11



CIRCULAR QUEUE OF  
"SCORES", OR FINISH TIMES, etc.

FIG. 5B

8/11

## UPDATE "FREE GAME TIME"

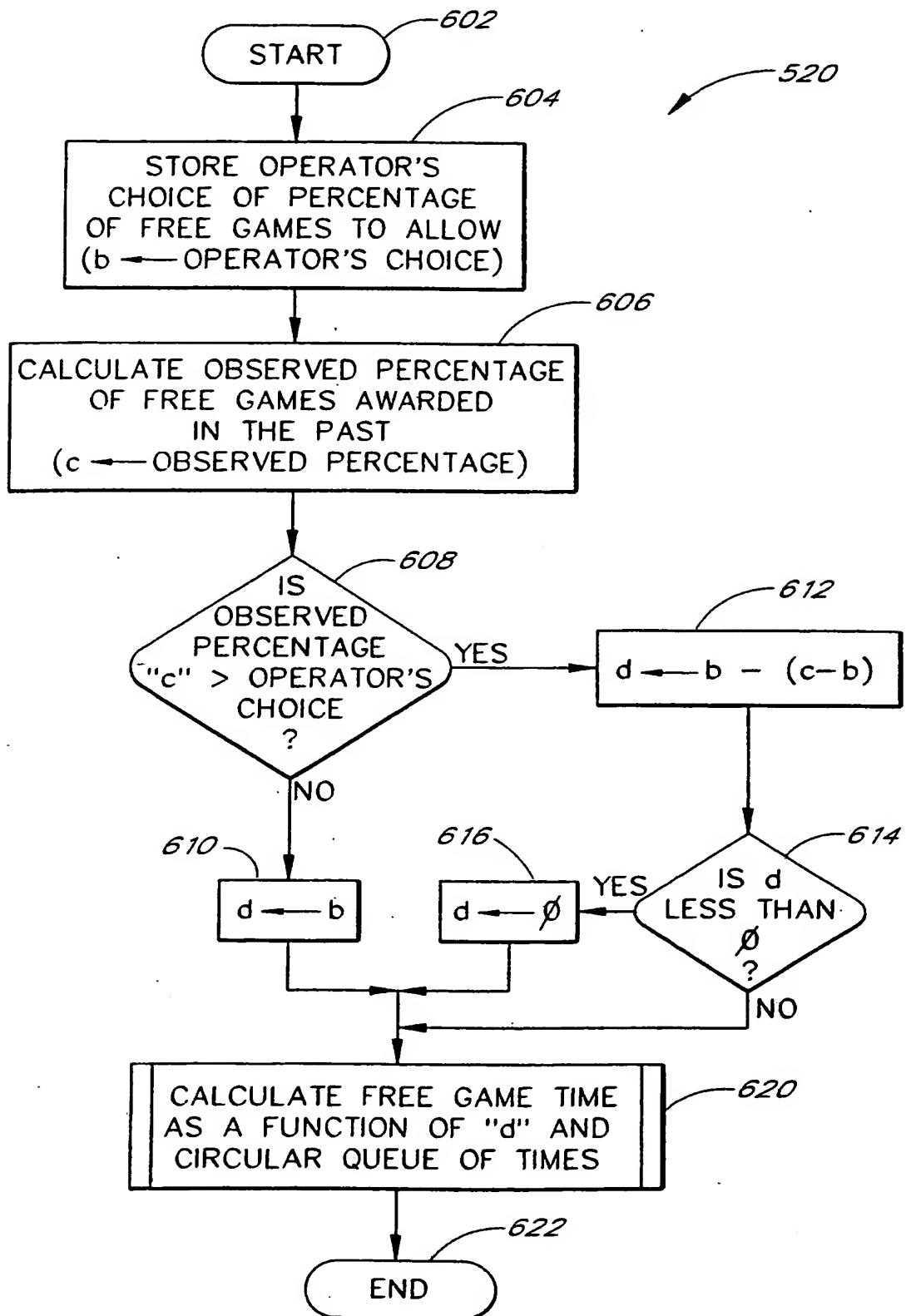


FIG. 6

9/11

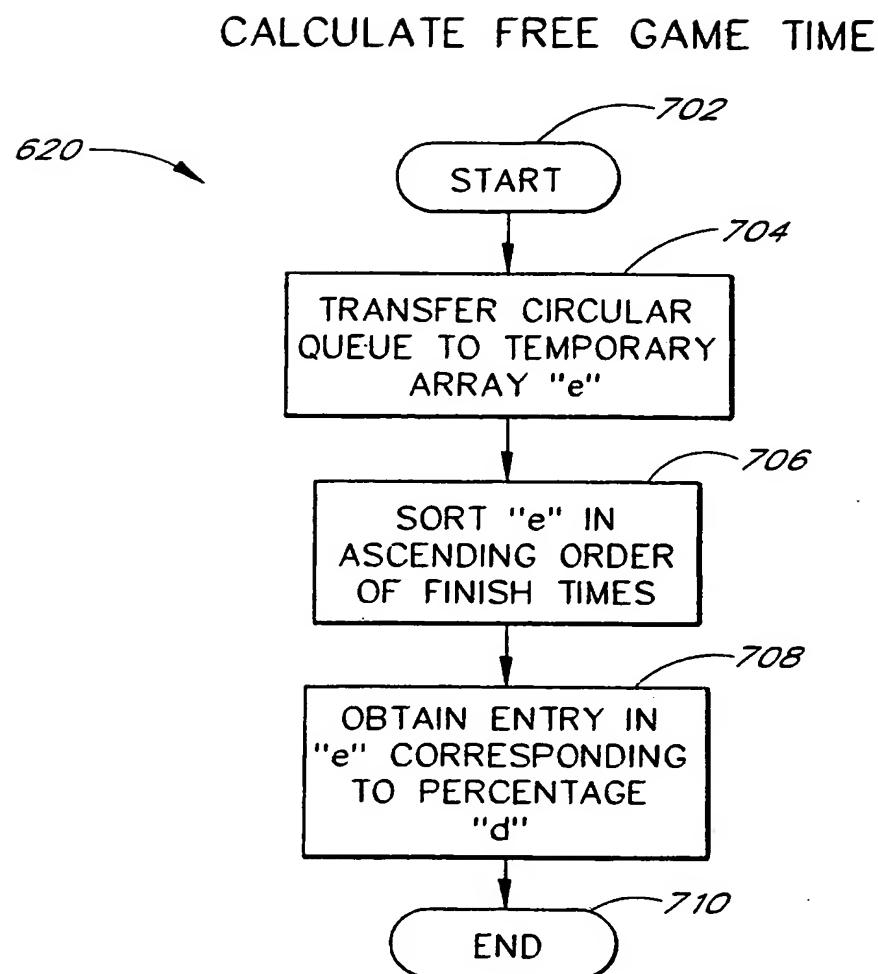


FIG. 7

10/11

## SAVING "FREE GAME" GHOST DATA

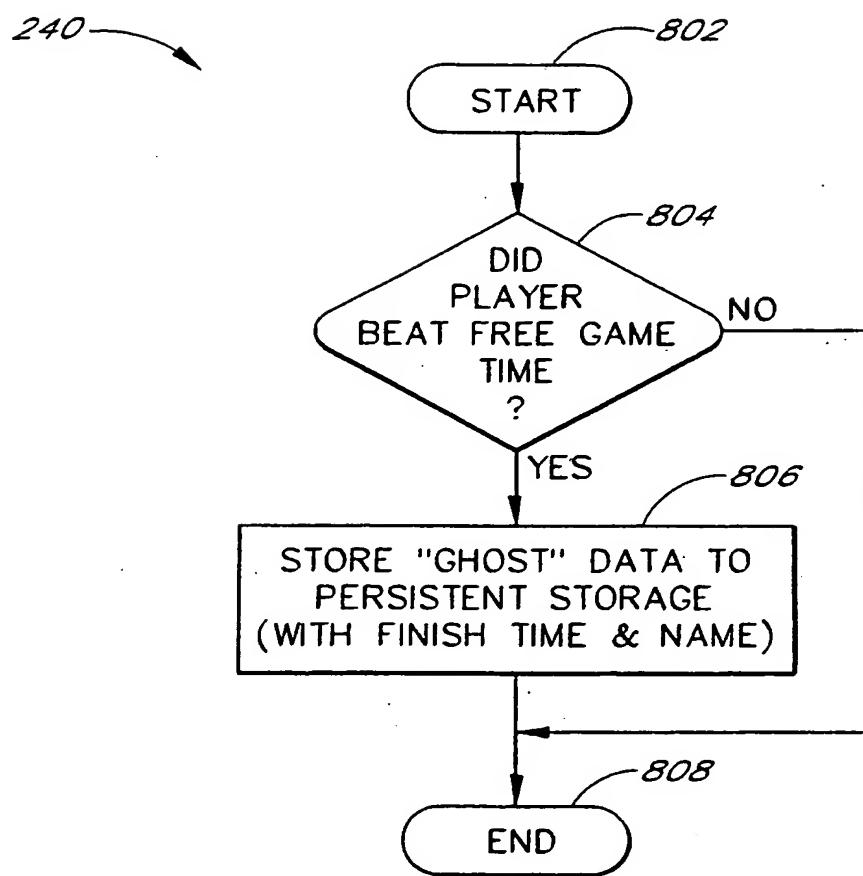


FIG. 8

11/11

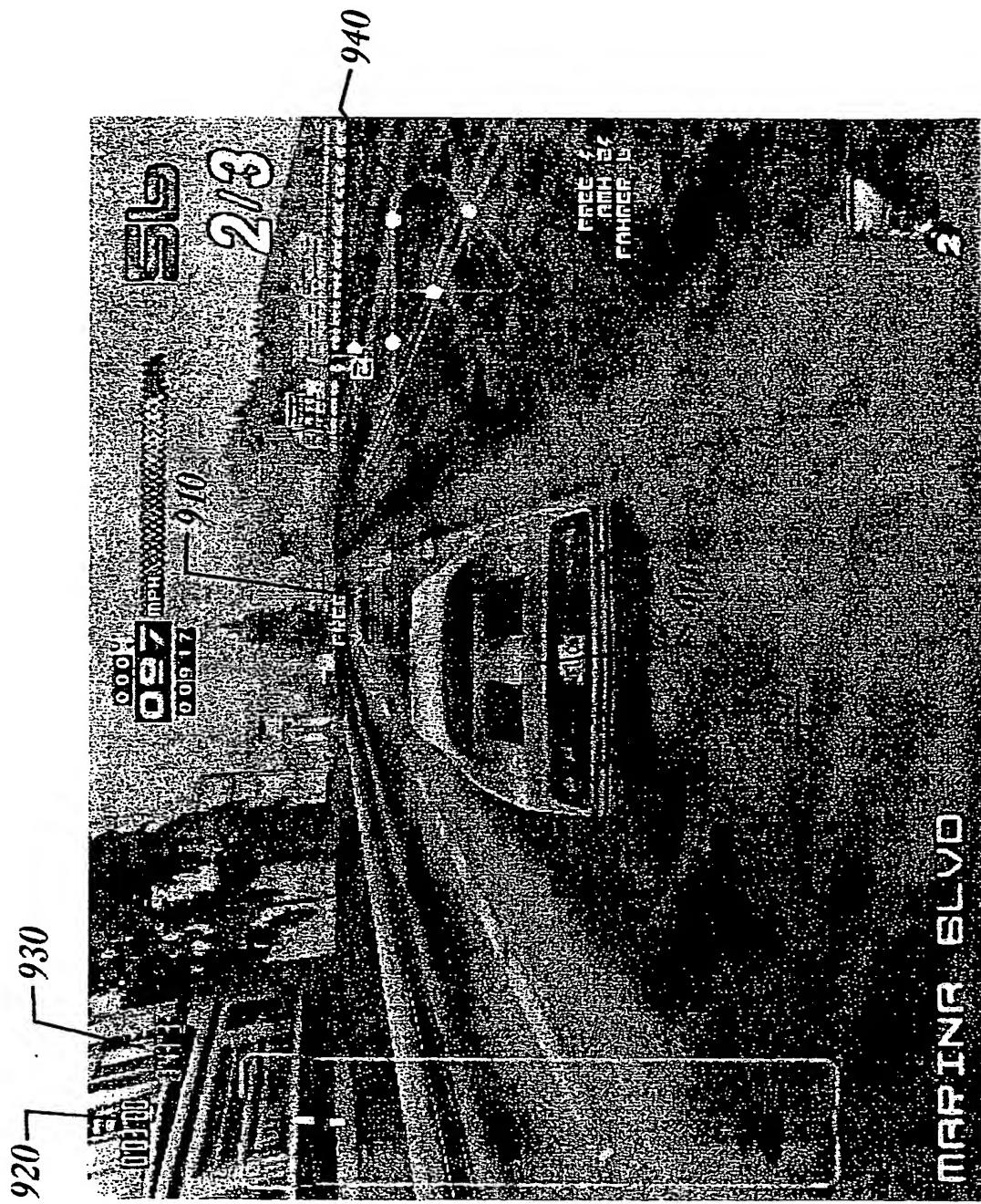


FIG. 9

BEST AVAILABLE COPY

## INTER' TIONAL SEARCH REPORT

Int'l

Final Application No

PCT/US 00/19352

A. CLASSIFICATION OF SUBJECT MATTER  
IPC 7 A63F13/10

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)  
IPC 7 A63F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 836 873 A (SEGA ENTERPRISES KK) 22 April 1998 (1998-04-22)	6, 8, 23, 24, 26, 27, 30, 31, 36 1-5, 9, 10, 13-21, 25, 29, 37-39
Y	column 4, line 10 - line 27 column 11, line 58 -column 12, line 3 column 12, line 34 - line 45 column 13, line 5 - line 12 column 13, line 36 - line 54 column 14, line 2 - line 5 column 17, line 1 - line 20 ----	-/-

 Further documents are listed in the continuation of box C. Patent family members are listed in annex.

\* Special categories of cited documents :

- "A" document defining the general state of the art which is not considered to be of particular relevance
- "E" earlier document but published on or after the international filing date
- "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- "O" document referring to an oral disclosure, use, exhibition or other means
- "P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search

Date of mailing of the international search report

12 October 2000

18/10/2000

Name and mailing address of the ISA  
European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl.  
Fax: (+31-70) 340-3016

Authorized officer

Sindic, G

## INTERNATIONAL SEARCH REPORT

Int'l Application No
PCT/JS 00/19352

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 4 685 677 A (DEMAR LAWRENCE E ET AL) 11 August 1987 (1987-08-11)	35
Y	column 2, line 39 - line 54 column 6, line 7 - line 68 _____	1-5, 9, 10, 13-21, 25, 29, 37-39
A	US 5 354 202 A (BEHENSKY MAX L ET AL) 11 October 1994 (1994-10-11)  column 10, line 18 - line 45 column 10, line 65 -column 11, line 12 _____	1, 2, 6, 21, 23, 35, 36, 39
A	US 5 366 376 A (GRAY ALAN S ET AL) 22 November 1994 (1994-11-22) column 9, line 44 - line 59 column 10, line 7 - line 40 _____	1, 2

**INTERNATIONAL SEARCH REPORT**

Information on patent family members

Int. Application No  
PCT/US 00/19352

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP 0836873 A	22-04-1998	CN WO	1190353 A 9741935 A	12-08-1998 13-11-1997
US 4685677 A	11-08-1987	AT AU AU BR CA DE DE EP JP	83087 T 590178 B 7522487 A 8703264 A 1276305 A 3782868 A 3782868 T 0252590 A 63024976 A	15-12-1992 26-10-1989 14-01-1988 15-03-1988 13-11-1990 14-01-1993 08-07-1993 13-01-1988 02-02-1988
US 5354202 A	11-10-1994	US AT CA CA DE EP JP US WO WO	5577913 A 138486 T 2067093 A 2067132 A 69119739 D 0495083 A 5501981 T 5269687 A 9202917 A 9202916 A	26-11-1996 15-06-1996 02-02-1992 02-02-1992 27-06-1996 22-07-1992 15-04-1993 14-12-1993 20-02-1992 20-02-1992
US 5366376 A	22-11-1994	EP JP US WO US US US US	0641471 A 7507402 T 5368484 A 9324915 A 5607308 A 5618178 A 5618179 A 5573402 A	08-03-1995 10-08-1995 29-11-1994 09-12-1993 04-03-1997 08-04-1997 08-04-1997 12-11-1996